# PACKET SWITCH WITH ENHANCED FLOW CONTROL

## CROSS-REFERENCE TO RELATED APPLICATIONS

**[0001]** Not Applicable

## STATEMENT OF FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

**[0002]** Not Applicable

## BACKGROUND OF THE INVENTION

### 1. TECHNICAL FIELD

**[0003]** This invention relates in general to packet switches and, more particularly, to a method and apparatus for efficiently controlling flows between queues in a packet switch.

### 2. DESCRIPTION OF THE RELATED ART

**[0004]** Packet switches (also referred to as "routers") are used throughout communications networks to route information. Typically, a packet switch has input and output ports, with respective input and output queues, to temporarily store data passing through the switch. In order to reduce the probability of overflow, flow control logic is generally required.

**[0005]** Present day flow control logic controls either the absolute quantity of data that can be sent from the input port to the output port or the rate at which data can be sent from input port to output port.

**[0006]** A complicating factor in flow control logic is the time delay in the delivery of the control packets, which regulate the flow. Delivery of flow control packets through the switch slows down the available data flow and, hence, it is preferable to use a few control packets as possible. On the other hand, using long times between data packets increases the chance that the information in the control packet may be out of date. Out of date control information may lead to changes in the flow between input and output queues which increase congestion.

**[0007]** Accordingly, a need has arisen for a packet switch with efficient flow control logic.

BRIEF SUMMARY OF THE INVENTION

**[0008]** In the present invention, a packet switch, comprises one or more output queues for temporarily storing packets to be output from the packet switch and one or more input queues for temporarily storing received packets. Each input queue is associated with an output queue. A matrix passes information between said inputs queues and said output queues. Control signals are generated to control the change of a transfer rate between associated input queues and output queues.

**[0009]** The present invention provides significant advantages over the prior art. By controlling the change in the transfer rate, the flow of packets can be more efficiently controlled, using fewer control cycles.

## BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

**[0010]** For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

**[0011]** Figure 1 illustrates a block diagram of a network;

**[0012]** Figure 2 illustrates a block diagram of a packet switch;

**[0013]** Figure 3 illustrates a more detailed block diagram of the packet switch of Figure 2;

**[0014]** Figure 4 illustrates the flow of backward force control messages from an output port to an input port of the packet switch of Figure 3;

**[0015]** Figure 5 illustrates the flow of forward force control messages from an input port to an output port of the packet switch of Figure 3;

**[0016]** Figure 6 illustrates a server for controlling the service rate of an input port;

**[0017]** Figure 7 illustrates a block diagram of a packet switch using multi-level input and output ports;

**[0018]** Figure 8 illustrates a block diagram of a input port using an input flow controller; and

**[0019]** Figure 9 illustrates a network using inter-switch control messages to control data flow between packet switches.

DETAILED DESCRIPTION OF THE INVENTION

**[0020]** The present invention is best understood in relation to Figures 1 - 9 of the drawings, like numerals being used for like elements of the various drawings.

**[0021]** Figure 1 illustrates a basic diagram of a network 10 using packet switches 12 to route data packets over lines 14. The packet switches route incoming packets to a selected destination. The actual configuration of the network 10 is for example only; an actual network 10 would typically be much more complicated than that shown in Figure 1.

**[0022]** Typically, at least some of the packets received by a packet switch 12 do not pass directly to an output. These packets are temporarily stored in the packet switch in a memory or "queue". If a queue becomes full, additional incoming packets will cause some packets to be dropped. The dropped packets may be the newly received packets or packets already in the queue, depending upon the implementation of the packet switch 12. Naturally, it is desirable to drop as few packets as possible.

**[0023]** Figure 2 illustrates a basic diagram of a packet switch 12. Each packet switch 12 is connected to one or more input lines 14. Packets from these lines are received at the input ports 16, which include the input queues (see Figure 3). A matrix 18 allows a packet from any of the input ports 16 to a selected output port 20. The output ports 20 are coupled to the lines 14 to send packets to other packet switches 12.

**[0024]** Figure 3 illustrates a more detailed block diagram of a packet switch 12. In the illustrated embodiment, the packet switch 12 has $x+1$ input ports 16 (individually shown as output ports $16_0$ through $16_x$) and $y+1$ output ports 20 (individually shown as output ports $20_0$ through $20_y$). Each input port 16 includes up to $y+1$ memory queues 22, each queue corresponding to a respective

output port 20. Each output port 20 has an output queue 24. For reference, individual input queues are shown as $IQ_{port,queue}$ and individual output queues are shown as $OQ_{port}$.

**[0025]** In operation, as packets are received by an input port 20, each packet is placed in the queue 22 corresponding to the desired output port 20. Under control of the input port 16, a packet at the front of a selected input queue 22 will pass through the matrix 18 to the corresponding output queue 20 on each matrix cycle. The output queues 20 will transmit packets to outgoing lines 14, but possibly not as fast as the packets are received from the input ports 16.

**[0026]** Hence, if a large number of received packets are directed to a particular output port 20, the input queues 22 corresponding to that output port 20 will tend to be more occupied than the queues corresponding to other output ports. Similarly, the output queue 24 corresponding to a popular output port 20 can be more occupied than less popular output queues 24. If either the input queues 22 or output queues 24 become full, packets will be dropped, which may cause the source of the packets to resend the packets.

**[0027]** Figures 4 and 5 illustrate an embodiment of the invention, where the input ports 16 and output ports 20 send forward and backward "force" control messages to one another based on criteria to be discussed hereinbelow. Each input port 16 includes logic 26 for calculating the value of the forward force control messages and each output port 20 includes logic 28 for calculating the value of the backward force control values. Logic 26 could be centralized in each input port 16 for all input queues of an input port or logic 26 could be distributed among the various input queues 22 for an input port 16; i.e., each input queue could have its own logic 26. For purposes of illustration, force control values are shown as being sent from input queues 22 to output queue 24 (forward force messages) or output queues 24 to input queues 22 (backward force messages); however, the path of the control messages could shown as between logic 26 and

logic 28. The input ports 16 also may include servers 30 that control transmission of packets into the matrix 18, as described in greater detail in connection with Figure 6.

**[0028]** Figure 4 illustrates the preferred embodiment for the force backward (FB) control value. Each output queue 24 outputs a FB control message having the same value to all of the corresponding input queues 22. Hence the output queue 24 of output port $20_0$ outputs $FB_0$ to all queues $IQ_{x,0}$ and output port $20_y$ outputs value $FB_y$ to all input queues $IQ_{z,y}$, for all integer values of $z$ between 0 and $x$.

**[0029]** As shown in Figure 5, the input ports 20, however, generate a force forward (FF) control value for each individual input queue 22, which is sent to the output logic of the respective output port 20. Each FF control value is designated $FF_{port,queue}$. Hence the FF control value from the input queue of input queue $IQ_{0,1}$ would be designated as $FF_{0,1}$.

**[0030]** Importantly, the values generated in the present invention control the acceleration of the rate of data packets between a particular input queue 22 and a particular output queue 24. This has been found effective in controlling the occupancy of the queues 22 and 24, while reducing the amount of matrix resources used to transfer the control messages with respect to data packets.

**[0031]** Various methods can be used to generate the force values. In one embodiment, the following configuration parameters are used:

$Li_{i,j}$ is the length of occupied portion of input queue j on port $i$.

$Lo_j$ is the length of occupied portion of output queue $j$.

$Li_{max}$ and $Lo_{max}$ are the maximum lengths of the input and output queues.

$age_{i,j}$ is the age of the packet at the head of input queue $j$ on port $i$.

$age_{max}$ is the maximum allowable age of a packet in the input queue

before it is discarded.

$w_1$ and $w_2$ are weights that represent the relative importance of the length of an input queue and the age of the packet at the head of the input queue, $w_1 >= 0$, $w_2 >= 0$, and $w_1 + w_2 = 1$

**[0032]** Using these configuration parameters, a force forward value can be calculated as:

$$FF_{i,j} = w_1 ( Li_{i,j} / Li_{max} ) + w_2 ( age_{i,j} / age_{max} ),$$

where $FF_{i,j}$ is calculated at input port i and $0 <= FF_{i,j} <= 1$.

**[0033]** A backward force value is calculated as:

$$FB_j = g_b ( Lo_j / Lo_{max} - \tfrac{1}{2} ) 2 + ( ( FF_{0,j} + FF_{1,j} \ldots + FF_{x,j} ) / (x+1) ),$$

where $x+1$ is the number of input ports and $g_b$ is a positive constant that represents the feedback gain. $FB_j$ is calculated at output port $j$ and $-g_b <= FB_j <= g_b + 1$.

**[0034]** For each input queue 22 there are four values named net force (FN), mass ($m$), acceleration ($a$), and service rate ( $r$ ). Net force depends on forward force and backward force:

$$FN_{i,j} = lpf( FF_{i,j} ) - FB_j,$$

where lpf() is a low pass filter that serves as a damping filter. In a practical implementation, lpf() could be a running average.

**[0035]** Acceleration depends on mass and net force:

$$a_{i,j} = FN_{i,j} / m_{i,j},$$

where mass is a configured attribute of an input queue. The acceleration is the derivative of the service rate:

$$a_{i,j} = r_{i,j}'.$$

**[0036]** In the preferred embodiment, forward force and backward force messages are sent on a regularly scheduled basis. Every time an input queue 22

receives a backward force message, it recalculates $a_{i,j}$ and adds the current $a_{i,j}$ to the previous value of $r_{i,j}$. The service rate (described below), $r_{i,j}$, is constrained such that, $0 <= r_{i,j} <= 1$.

**[0037]** Figure 6 illustrates the operation of a server 30 for an arbitrary input port $16_z$. Typically, every input queue 22 on the input port 16 cannot send packets to the matrix 18 at the same time. Once each "matrix" cycle, the matrix 18 passes data packets between input ports 16 and output ports 20. On a "control" cycle, control messages are passed between input ports 16 and output ports 20. It is desirable to have a high ratio of matrix cycles to control cycles.

**[0038]** In the following discussion, it is assumed that (1) internal packets are all of the same fixed size, (2) the matrix operates in discrete cycles, and (3) an input port 16 can only send one packet from one input queue 22 in one matrix cycle; this is known as having one server per input port.

**[0039]** The server 30 is a logic circuit that connects one of input queues 22 of an input port 16 to the matrix 18 at a time. When an input queue 22 is not empty, it makes a request to the server. Each cycle, the server 30 picks one input queue 22 from among all the input queues 22 that have made requests and connects that queue to the matrix 18. The chosen input queue 22 sends one packet. The server 30 can be designed not to favor one queue over another. For example, it may select input queues 22 in round-robin fashion. If an input port 16 has more than one connection to the matrix 18, it is possible to have more than one server per port.

**[0040]** As described above, the service rate, $r_{i,j}$, is an attribute of an input queue, and it is a number between 0 and 1. In the model above, the service rate controls the probability that a non-empty queue will make a request to the server. So, if an input queue has a service rate of 0.5 and it is non-empty, there is

a 50% chance the queue will make a request for service in the current matrix cycle.

**[0041]** In the equations above, the net force (FN) tends to be zero when the output queue is half full. When an output queue 24 is more or less than half full, its associated net force tends to be negative or positive, respectively. This is an imperfect tendency because of transmission delays and the low pass filter applied to the forward force function. A balance point can be defined as the length of the output queue at which net force and acceleration tend to be zero. Typically, using the calculations for FF, FB and FN described above, the balance point of an output queue is 0.5.

**[0042]** It is possible to modify the backward force equation such that the balance point is a configuration option. If $b$ is the balance point of the output queue and $0 <= b <= 1$:

$$FB_j = g_b\,(Lo_j / Lo_{max} - b)(1/(1-b)) + ((FF_{0,j} + FF_{1,j} \ldots + FF_{x,j}) / (x+1)),$$

where $x+1$ is the number of input ports.

**[0043]** The larger the balance point the more full the output queue 24 tends to be during normal operation and the less full the respective input queue 22 tends to be. If the balance point is greater than zero, the service rate tends to be "1" under lightly loaded conditions. This allows for minimal latency in a lightly loaded situation. Giving different output queues different balance points can give them different qualities of services.

**[0044]** The operation of the network 10 can also be modified by adjusting the mass of an input queue 22. Increasing the mass of an input queue 22 decreases the variance of its output data rate. It also reduces the variance of the output data rate of the corresponding output queue 24. A higher mass for an input queue 22 also decreases the average length of the output queue 24 and increases the average length of the input queue 22.

**[0045]** Decreasing the variance of the output data rate of the output queues is desirable because it reduces the general of burstyness of all traffic in the network. This benefits downstream routers.

**[0046]** It is possible to modify the scheme described above such that each input queue 22 has two masses *m1* and *m2*. Acceleration is calculated as follows:

If $FN_{i,j} > 0$ then $a_{i,j} = FN_{i,j} / m1_{i,j}$

If $FN_{i,j} <= 0$ then $a_{i,j} = FN_{i,j} / m2_{i,j}$

Generally $m2_{i,j}$ would be much smaller than $m1_{i,j}$.

**[0047]** The advantage to this scheme is that when the output queue 24 begins to become full, it can throttle the input queues at a faster rate. This reduces the probability that the output queue 24 will overflow.

**[0048]** Figure 7 illustrates a multi-stage packet switch 32, in which there can be two or more levels of input queues 16 (shown as outer input queues 16a and inner input queues 16b) and two or more levels of output queues 20 (shown as outer output queues 20a and inner output queues 20b). Decreasing the variance of the output data rate of the input queues 16 is also desirable in a multi-stage switch. In such a system the force messages would be exchanged between the outer input queues 16a and outer output queues 20a only.

**[0049]** The inner input queues 16b and inner output queues 20a have higher data rates than the outer queues 16a and 20a. For this reason the inner queues 16b and 20b are more expensive than the outer queues. There is a cost advantage to keeping the lengths of the inner queues 16a and 20a small. A small variance in the data rate of the data entering the inner input queues 16a has this effect.

**[0050]** Mass reduces the rate at which the output data rate of an input queue 16 can change. For this reason, control messages can be less frequent. Since control messages consume resources in the matrix 18, this increases the fraction

of the matrix capacity that can be devoted to carrying payload. Increasing the mass of an input queue 16 has the same effect as proportionally decreasing the forward and backward forces acting on that queue.

**[0051]** Instability in the packet switch 12 can be addressed by the damping filter function. Changing the damping filter function, lpf(), has an effect similar to changing a queue's mass. A stronger damping filter reduces the probability of an output queue overflowing, but it increases the probability of an input queue overflowing. A stronger damping filter also reduces the variance of the output data rates of both the input queues 22 and output queues 24. Like mass, the damping filter reduces the rate at which the output data rate of an input queue 22 can change. This increases the fraction of the matrix capacity that can be devoted to carrying payload.

**[0052]** In some embodiments of a packet switch 12, different input queues 16 may be given different configuration parameters pursuant to different quality of service associated with the input queues. For example, jitter occurs when time interval between two packets is not the same when they are received, as it was when they were sent. Jitter is a particular problem for streaming audio and video. A flow of packet between two host systems can increase in jitter as it passes through a switch 12.

**[0053]** An input queue 16 (or multiple input queues) in a switch 12 can be configured to have less jitter by leaving the configuration of all other queues the same and modifying the configurations of this queue in the following ways:

1) Increase its forward force by multiplying its forward force by a scaling constant; and/or

2) Change its forward force weights to favor packet age over queue length.

**[0054]** Figure 8 illustrates an alternative embodiment of the invention where an input flow controller 34 is used in each input port 16. The input flow controller 32 generates a force FC. In this case, acceleration would be equal to:

$$a_{i,j} = \text{lpf}( FF_{i,j} ) - ( w_1 FB_j + w_2 FC_i ),$$

where $w_1$ and $w_2$ are weights that determine the relative importance of backward force and input flow controller force. Further,

$$FC_i = g_c (( r_{i,0} + r_{i,1} + \ldots + r_{i,x})/ R_i - \tfrac{1}{2} ) 2 + ((FF_{i,0} + FF_{i,1} \ldots + FF_{i,x})/(x+1))$$

where $R_i$ is the maximum bandwidth an input port can send toward the matrix, and $g_c$ is a constant that represents the gain on the input flow controller's feedback.

**[0055]** It may be desirable to reduce sudden changes in the service rate. One method of reducing sudden changes is to use a non-constant service rate. In one matrix cycle, payload data packets are sent from input queues 22 to output queues 24. In one control cycle, forward force packets are sent from input queues 22 to output queues 24, and backward force packets are sent from output queues 24 to input queues 22. In the scheme described above, it assumed that there are multiple matrix cycles for every control cycle. As described, the service rate, $r_{i,j}$, is changed once every control cycle, and remains constant for multiple matrix cycles.

**[0056]** The problem with this the approach is that it causes sudden changes in service rate. This causes burstyness in the input queues output data rate. This burstyness becomes larger as the number of matrix cycles per control cycle becomes larger. It is desirable to have a large number of matrix cycles per control cycle, because this reduces the fraction of the matrix capacity that is consumed by control packets. In some instances, it may be preferable to allow the service rate to vary during a single control cycle.

**[0057]** The simplest way to allow the service rate to vary during a control cycle is to apply the same acceleration to the service rate every matrix cycle. Assume the control cycles within a matrix cycle are numbered from 1 to N where N is the number of matrix cycles per control cycle. Further assume that at the end of the previous control cycle, the acceleration for input queue i,j was $a_{i,j}$, and for any control cycle, k, within the matrix cycle (where 1 <= k <= N) the service rate for input queue *i,j* is $r_{i,j,k}$, let:

$$r_{i,j,k} = r_{i,j,k-1} + ( a_{i,j} / N ).$$

**[0058]** A more sophisticated approach is to calculate a new net force every matrix cycle, and use the net force value calculated at the end of a control cycle only as an upper bound for the per-matrix-cycle net force. As described above, the backward force and net force calculated at the end of a control cycle are $FB_j$ and $FN_{i,j}$. Let the forward force calculated any matrix cycle, k, be $FF_{i,j,k}$, let the net force calculated that matrix cycle be $FN_{i,j,k}$, and let the acceleration calculated that matrix cycle be $a_{i,j,k}$. Every matrix cycle the new net force is:

$$FN_{i,j,k} = \min( FN_{i,j}, lpf( FF_{i,j,k} ) - FB_j ).$$

The new acceleration is:

$$a_{i,j,k} = FN_{i,j,k} / m_{i,j}$$

And the new service rate is:

$$r_{i,j,k} = r_{i,j,k-1} + ( a_{i,j,k} / N ).$$

**[0059]** An alternative formula for the per-matrix-cycle net force leaves off the low pass filter:

$$FN_{i,j,k} = \min( FN_{i,j}, FF_{i,j,k} - FB_j ).$$

**[0060]** The advantage a non-constant service rate is a reduction in sudden changes in service rate, which causes burstyness in the input queues output data rate. The disadvantage is that this approach is more complicated. The advantage of recalculating the net force every matrix cycle is that it allows the

service rate of an input queue to grow more slowly or decline faster during a control cycle. This allows the server to give resources to other queues.

**[0061]** Another variation would be to set the service rate as a constant times the net force:

$$r_{i,j} = FN_{i,j} * k_{i,j},$$

where $k_{i,j}$ is a constant.

**[0062]** With this scheme, the service rate does not change during a control cycle. The advantage to this scheme is that it is simpler, and simulations show that it does a good job of controlling the lengths of the input and output queues.

**[0063]** The constant $k_{i,j}$ can be made asymmetric such that:

If $FN_{i,j} > 0$ then $a_{i,j} = FN_{i,j} * k1_{i,j}$

If $FN_{i,j} <$ or $= 0$ then $a_{i,j} = FN_{i,j} * k2_{i,j}$

**[0064]** It is possible to modify the scheme described above to have only backward force packets. In this approach forward force messages are eliminated and the formula for calculating backward force becomes:

$$FB_j = g_b ( Lo_j / Lo_{max} )$$

**[0065]** In this scheme the output queue 24 has no balance point. Net force tends to be zero when the input queues and the output queue are the same length (provided $g_b = 1$ and forward force does not depend on packet age). The advantage to this approach is that it is simpler and uses less matrix capacity. The disadvantage is that it does not do as good a job of reducing the variance of the output data rate of the input queue.

**[0066]** The present invention provides significant advantages over the prior art. By controlling the acceleration of the rate of packets between a input queue and an output queue, a desired balance can be maintained, without a heavy overhead of control messages through the matrix.

**[0067]** Figure 9 illustrates an embodiment of a network 40, where packet switches 12 use the concepts described above for controlling the flow of packets between input and output queues of different packet switches 12, rather than between input and output queues of the same packet switch. In this embodiment, an output queue 24 of a packet switch 12 will send a force forward control message to an input queue 22 of a different packet switch 12. Similarly, the input queue 22 will send a force backward control message to the associated output queue. The calculation and handling of the inter-switch messages can have the same properties as those of the intra-switch messages described above.

**[0068]** Although the Detailed Description of the invention has been directed to certain exemplary embodiments, various modifications of these embodiments, as well as alternative embodiments, will be suggested to those skilled in the art. The invention encompasses any modifications or alternative embodiments that fall within the scope of the Claims.